

APPENDIX C

Pseudo Code

The pseudo code below illustrates how ModelService calls the other components to implement its functionalities.

```
public class ModelService : IModelService
{
    // various work modules
    private IMapLoader mapLoader = null;
    private IMapWalker mapWalker = null;
    private IModelGenerator modelGenerator = null;
    private IModelMaterializer modelMaterializer = null;
    private ICodeGenerator codeGenerator = null;

    // inputs
    private ArrayList mapFiles = new ArrayList();

    // configuration
    private Config config = null;
    private Hint hint = null;

    // intermediate results
    private EntityMapCollection maps = null;
    private DataSet schema = null;
    private UDMModel udmModel = null;

    // instantiate a session of ModelService using the default work modules
    public ModelService()
    {
        mapLoader = new MapLoader();
        mapWalker = new MapWalker();
        modelGenerator = new ModelGenerator();
        modelMaterializer = new ModelMaterializer();
        codeGenerator = new CodeGenerator();
    }

    // set the configuration object for this ModelService session
    public void SetConfig(Config config)
    {
        this.config = config;
        GetHintFromConfig();
    }

    // obtain hint information from the hint file declared in config if any
    public void GetHintFromConfig()
    {
        this.hint = Hint.Deserialize(config.HintFileName);
    }

    // add a map file for later processing
    public void AddMapFile(string fileName)
    {
        mapFiles.Add(fileName);
    }
}
```

```

    }

    // this is where the real processing goes
    public void Process()
    {
        // 1. LOAD THE MAPS
        //
        LoadMaps();

        // 2. WALK THE MAPS
        //
        WalkMaps();

        // 3. GENERATE THE UDM MODEL
        //
        GenerateModel();

        // 4. MATERIALIZE THE UDM MODEL
        //
        MaterializeModel();

        // 5. GENERATE THE BIENTITY CODE FOR ACCESS TO THE UDM MODEL
        USING THE FRAMEWORK
        //
        GenerateCode();

        // 6. PROCESS THE UDM MODEL
        //
        ProcessModel();
    }

    // invoke MapLoader to load maps
    public void LoadMaps()
    {
        mapLoader.SetMapTransformFile(this.config.MapTransformFileName);

        // add each map file to the map loader and ask it to load the
        maps
        foreach(string mapFile in this.mapFiles)
        {
            mapLoader.AddMapFile(mapFile);
        }
        mapLoader.LoadMaps();

        // retrieve the collection of loaded maps from map loader
        this.maps = mapLoader.EntityMaps;
    }

    // invoke MapWalker to walk maps
    public void WalkMaps()
    {
        // configure the mapWalker
        mapWalker.SetDBSchemaName(this.config.DbSchemaName);
        mapWalker.SetMeasureHints(this.hint.MeasureHints);

        // pass the maps obtained from MapLoader to be processed
    }

```

```

        mapWalker.SetEntityMapCollection(this.maps);

        // walk the maps using the MapWalker and retrieve the result
        mapWalker.WalkEntityMaps();
        this.schema = mapWalker.Schema;
    }

    // invoke ModelGenerator to generate UDM Model
    public void GenerateModel()
    {
        // configure the modelGenerator
        modelGenerator.SetDataSource(this.config.DbServerName,
this.config.DbDatabaseName);
        modelGenerator.SetHint(this.hint);

        // pass the schema to ModelGenerator to be processed
        modelGenerator.SetSchema(this.schema);

        // generate the UDM model and retrieve the result
        modelGenerator.Generate();
        this.udmModel = modelGenerator.UdmModel;
    }

    // invoke ModelMaterializer to materialize UDM model
    public void MaterializeModel()
    {
        modelMaterializer.SetUDMServerName(this.config.UdmServerName);
        modelMaterializer.SetLogFile(this.config.UdmLogFileName);

        modelMaterializer.SetDropAllDatabases(this.config.DropUdmDatabases);

        // pass the udmmodel to be materialized to the ModelMaterializer
        modelMaterializer.SetUdmModel(this.udmModel);

        // materialize the result to the UDM server
        modelMaterializer.Materialize();
    }

    // invoke CodeGenerator to generate BIEntity code
    public void GenerateCode()
    {
        // set inputs to code generator
        codeGenerator.SetUdmModel(this.udmModel);
        codeGenerator.SetBICodeGenerator(null);

        // generate the code for BIEntity classes
        codeGenerator.Generate();
    }

    // invoke ModelProcessor to process UDM model generated
    public void ProcessModel()
    {
        modelMaterializer.Process();
    }
}

```